

# Instruction Fine-Tuning

Malay Agarwal

## Contents

<b>Limitations of ICL</b>	<b>1</b>
<b>Instruction Fine-Tuning</b>	<b>2</b>
Introduction . . . . .	2
Common Steps Involved in Instruction Fine-Tuning . . . . .	2
Prepare the Dataset . . . . .	2
Split Dataset . . . . .	3
Training . . . . .	3
<b>Fine-Tuning On a Single Task</b>	<b>4</b>
Catastrophic Forgetting . . . . .	4
Avoiding Catastrophic Forgetting . . . . .	4
<b>Fine-Tuning On Multiple Tasks</b>	<b>4</b>
Case Study - FLAN . . . . .	5
<b>Useful References</b>	<b>6</b>

## Limitations of ICL

We saw how some models are capable of identifying instructions contained in a prompt and correctly carrying out zero-shot inference. On the other hand, we also saw smaller models which fail to do so. In such cases, we use In-Context Learning (ICL) to make the model follow our instructions.

There are some disadvantages to this:

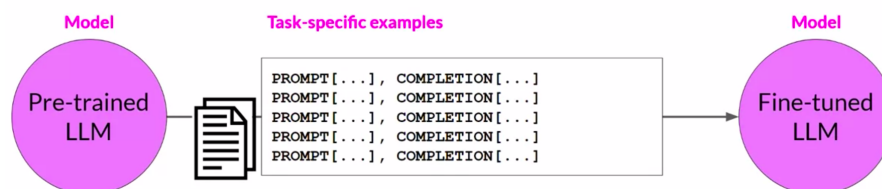
- ICL may not always work for smaller models.
- Examples take up space in the context window, reducing the space available to add useful information in the prompt.

To combat these disadvantages while having a model that can follow instructions, we can use instruction fine-tuning.

# Instruction Fine-Tuning

## Introduction

Fine-tuning is the process of using **labelled data** to adapt a pre-trained model to a specific task or tasks. The data consists of prompt-completion pairs. Note that fine-tuning is applied on a pre-trained model and is **supervised**, as opposed to self-supervised.



Instruction fine-tuning is a fine-tuning technique used to improve a model's performance on a variety of tasks. Here, the training samples are prompts containing instructions while the labels are the expected response of the model in order to follow that instruction.

**Example:** If we want to fine-tune a model to improve its summarization ability, the dataset will contain prompts which look like as follows:

**Prompt:** Summarize the following text (EXAMPLE TEXT)

**Completion:** Summarize the following text (EXAMPLE TEXT)  
(EXAMPLE COMPLETION)

Instruction fine-tuning where all of the model's weights are updated is called **full fine-tuning**. This results in a new version of the model with updated weights. Note that full fine-tuning requires enough memory and compute budget to store all the gradients, optimizer states and other components updated during training (see Efficient Multi-GPU Compute Strategies).

## Common Steps Involved in Instruction Fine-Tuning

There are some common steps involved in fine-tuning.

### Prepare the Dataset

There are many publicly available datasets that have been used to train previous generations of LLMs. Most of these datasets are *not* formatted as instructions.

Developers have built **prompt template libraries** that can be used to take existing datasets (for example, Amazon product reviews) and turn them into

instruction prompt datasets for fine-tuning.

Prompt template libraries include many templates for different tasks. For example:

Classification / sentiment analysis

```
jinja: "Given the following review:\n{{review_body}}\npredict the associated rating\n\ from the following choices (1 being lowest and 5 being highest)\n- {{ answer_choices\n\ | join('\n- ') }} \n|||\n{{answer_choices[star_rating-1]}}"
```

Text generation

```
jinja: Generate a {{star_rating}}-star review (1 being lowest and 5 being highest)\nabout this product {{product_title}}. ||| {{review_body}}
```

Text summarization

```
jinja: "Give a short sentence describing the following product review:\n{{review_body}}\n\ \n|||\n{{review_headline}}"
```

Notice how each of the templates has an instruction in it: *predict the associated rating*, *generate an x-star review* and *give a short sentence describing the following product review*.

The result is a prompt with an instruction and the example from the original dataset.

## Split Dataset

After the dataset is prepared, like any supervised problem, we split the dataset into training, validation and test sets.

## Training

The fine-tuning training loop is similar to any other supervised training loop:

- Pass the training data in batches to the model and obtain predictions.
- Calculate the loss. The output of an LLM is a probability distribution over the tokens available in the dataset. Thus, we can compare the probability distribution of the prediction with that of the label and use the standard cross-entropy loss to calculate the loss.
- Calculate some evaluation metric.
- Pass the validation data to the model and obtain predictions.
- Calculate the loss (optional) and the same evaluation metric.
- Backpropagate the loss to update the weights and repeat from the beginning as the next epoch.

After training is done, we can evaluate the final performance of the model by passing it the test data and measuring the evaluation metric on model predictions.

This process leads to a new version of the model, often called an **Instruct Model**. It tends to perform better at the tasks we have fine-tuned it for.

## Fine-Tuning On a Single Task

Fine-tuning on a single task can be done by simply using a single-task dataset. That is, all prompt-completion pairs in the dataset have the same basic instruction in them.

**Example:** Summarize the following text: (EXAMPLE TEXT) (EXAMPLE COMPLETION)

In most cases, only a small dataset (500-1000 examples) is required to achieve good performance on a single-task.

## Catastrophic Forgetting

Fine-tuning on a single task can lead to a problem called **catastrophic forgetting**. This happens since full fine-tuning changes the weights of the original LLM. This leads to great performance on the task we are fine-tuning for but can degrade performance on other tasks.

For example, a model fine-tuned for sentiment analysis might become very good at the task, but might fail on something like named entity recognition despite being performant on it before fine-tuning.

## Avoiding Catastrophic Forgetting

First, we have to figure out whether our model is actually impacted by the problem. For example, if we require reliable performance only on the single task we are fine-tuning for, we do not need to worry about catastrophic forgetting.

But, if we want the model to maintain its multi-task performance, we can perform fine-tuning on multiple tasks at the same time. This generally requires 50,000-100,000 examples across many tasks.

Another alternative is **Parameter Efficient Fine-Tuning (PEFT)**. PEFT preserves the weights of the original LLM and trains only a small number of task-specific adapter layers and parameters (see Parameter Efficient Fine-Tuning (PEFT)).

## Fine-Tuning On Multiple Tasks

In case of multiple tasks, the dataset contains prompt-completion pairs related to multiple tasks.

**Example:**

Summarize the following text:

Rate this review:

Translate into Python code:

Identify the places:

The model is trained on this mixed dataset to fine-tune on multiple tasks simultaneously and remove the risk of catastrophic forgetting.

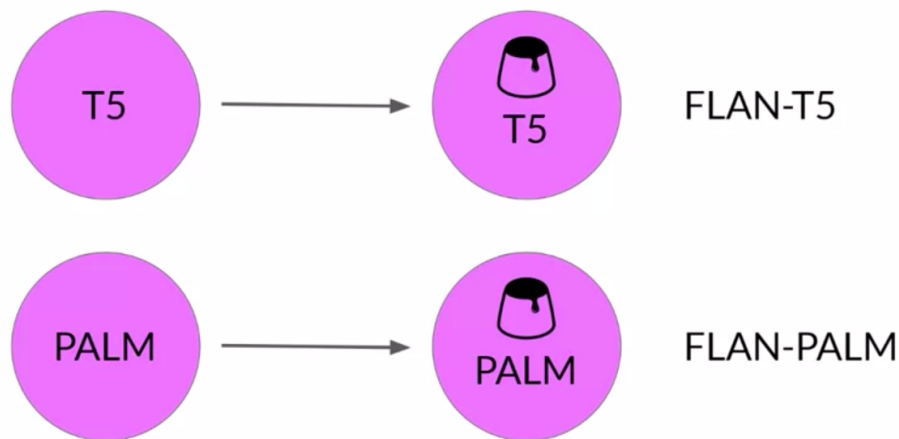
## Case Study - FLAN

FLAN (**F**ine-tuned **L**anguage **N**et) is a family of models fine-tuned on multiple tasks.

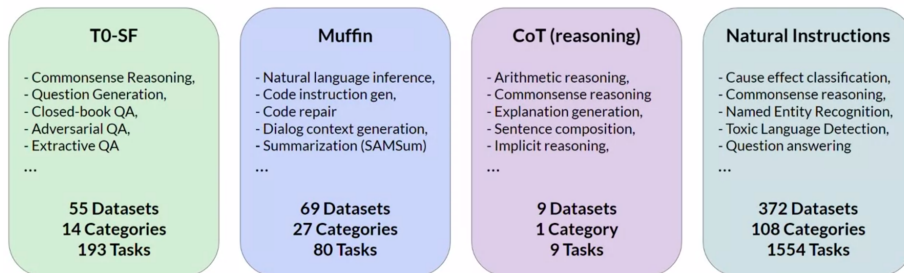
FLAN models refer to a specific set of instructions used to perform instruction fine-tuning.

The metaphorical dessert to the main course of pre-training.

FLAN-T5 is the FLAN instruct version of the T5 foundation model while FLAN-PALM is the FLAN instruct version of the PALM foundation model.






FLAN-T5 is general purpose instruct model. It is fine-tuned on 473 datasets across 146 task categories. These datasets are chosen from other models and papers.



For example, the SAMSum dataset is a text summarization dataset. SAMSum

has 16,000 messenger-like conversations with their summaries. They were crafted by linguists for the express purpose of training LLMs.

 <b>Datasets: samsun</b>	Tasks:  Summarization	Languages:  English
dialogue (string)	summary (string)	
"Amanda: I baked cookies. Do you want some? Jerry: Sure! Amanda: I'll bring you tomorrow :-)"	"Amanda baked cookies and will bring Jerry some tomorrow."	
"Olivia: Who are you voting for in this election? Oliver: Liberals as always. Olivia: Me too!! Oliver: Great"	"Olivia and Olivier are voting for liberals in this election. "	
"Tim: Hi, what's up? Kim: Bad mood tbh, I was going to do lots of stuff but ended up procrastinating Tim: What did..."	"Kim may try the pomodoro technique recommended by Tim to get more stuff done."	

Below are examples of prompt templates for this dataset.

```
"samsun": [
    ("{{dialogue}}\nBriefly summarize that dialogue.", "{{summary}}"),
    ("Here is a dialogue:\n{{dialogue}}\n\nWrite a short summary!",
     "{{summary}}"),
    ("Dialogue:\n{{dialogue}}\n\nWhat is a summary of this dialogue?",
     "{{summary}}"),
    ("{{dialogue}}\n\nWhat was that dialogue about, in two sentences or less?",
     "{{summary}}"),
    ("Here is a dialogue:\n{{dialogue}}\n\nWhat were they talking about?",
     "{{summary}}"),
    ("Dialogue:\n{{dialogue}}\n\nWhat were the main points in that "
     "conversation?", "{{summary}}"),
    ("Dialogue:\n{{dialogue}}\n\nWhat was going on in that conversation?",
     "{{summary}}"),
]
```

Note that while FLAN models are general-purpose, we might still need Domain Adaptation for it to make it work well for our application.

## Useful References

- FLAN paper.
- SAMSum paper.
- DialogSum on HuggingFace.